A fuzzy-XCS classifier system with linguistic hedges

Marin-Blázquez, Javier; Shen, Qiang

Published in:
Journal of Computational Intelligence Research (IJCIR)

Publication date:
2008

Citation for published version (APA):
Marin-Blázquez, J., & Shen, Q. (2008). A fuzzy-XCS classifier system with linguistic hedges. *Journal of Computational Intelligence Research (IJCIR)*, *4*(4), 329-341. http://hdl.handle.net/2160/2379

International Journal of Computational Intelligence Research.
ISSN 0974-1259 Vol.4, No.4 (2008), pp. 329–341

A Fuzzy-XCS Classifier System with Linguistic Hedges

Javier G. Marín-Blázquez[1] **and Qiang Shen**[2]

[1]Departamento de Ingeniería de la Información y las Comunicaciones
Facultad de Informática, Universidad de Murcia, Spain
jgmarin@um.es

[2]Department of Computer Science
Aberystwyth University, Wales, UK
qqs@aber.ac.uk

Abstract:
Many real-world problems require the development and application of algorithms that automatically generate human interpretable knowledge from historical data. Most existing algorithms for rule induction from imprecise data have followed the precise approach, where definitions of the fuzzy sets that are intended to capture certain vague concepts are allowed to be modified such that they fit the data. These approaches typically destroy the original semantics or meaning of the given fuzzy sets, which often leads to loss of transparency in the resulting model or models. In order to overcome this fundamental limitation, a descriptive approach has been proposed in which human defined fuzzy sets are not allowed to be modified. However, as the fuzzy set definitions cannot be modified, and only a small number of them are normally available, only a limited number of possible rules are derivable. Such rules are not very flexible and in many cases, will not necessarily fit the data well.
To address this important issue, at least partially, linguistic hedges have been introduced to provide a more adaptable means of learning from data, thereby offering more flexibility in domain knowledge representation and extraction. Following this approach, this paper presents a novel rule induction mechanism which extends a classifier system (XCS) by employing linguistic hedges. The resultant fuzzy XCS classifier with linguistic hedges is evaluated against a real-world forensic glass classification problem. The results demonstrate that the inclusion of hedges to support finer granularity in linguistic fuzzy modelling improves the accuracy of the resulting classifiers, whilst simultaneously preserving the interpretability of the learned models. This approach not only offers the user rules to decide on classes, but also rules to decide which classes to discard. It also inherits from XCS, the ability to deal with data that involves imbalanced classes.

I. Introduction

Amongst the contributions of Fuzzy Logic, perhaps one of the most fundamental is "computing with words" [34]. Despite the fact that there are considerable mathematical computations involved, the use of fuzzy rules allows the expression of imprecise dependencies with words in a very human-like manner. In terms of transparency, a fuzzy rule-based system is unrivalled by other approaches. A fuzzy set can be defined, and labelled, by humans, such that it describes their particular and subjective understanding of concepts of a particular domain. By employing such human defined and labelled sets embedded in simple production (aka. IF-THEN) rules, a powerful and clear form of knowledge representation can be achieved. Human experts can be the source of these rules, obtaining such expertise however has become an obstacle to building knowledge based systems, whether fuzzy or not.
Given the general increase in data available for many application problems, the development of algorithms that automatically generate, transparent, human readable knowledge from data is therefore highly desirable. In the past however, many of the proposed algorithms for fuzzy rule induction from data have followed the so-called precise approach. Interpretability is often sacrificed with such approaches, in exchange for a perceived increase in precision. In many cases the original fuzzy set definitions are modified in order to fit the data better. This modification comes at the cost of ruining the original meaning of the fuzzy sets and the loss of transparency of the resulting model. In other cases the algorithms generate the fuzzy sets, and present them to the user. The user must then interpret these sets and the rules which employ them. Furthermore, in some extreme cases, each rule may have its own fuzzy set definition for every condition, thereby generating many different sets in a modest rule base. The greatest

disadvantage of the precise approaches lies in the fact that the resulting sets and rules are difficult to match with the human interpretation of the relevant concepts.

As an alternative to the precise approach, there exist proposals that follow the linguistic (or descriptive) approach. In such an approach no changes are made to the human defined fuzzy sets. The rules must use the (fuzzy) words provided by the user without modifying them in any way. One of the main difficulties with this type of approach is that, as the fuzzy sets can not be modified, and only a small number of them are typically available, the possible rules available are predetermined, equivalently speaking. Although there can be many of these rules they are not very flexible and in many cases they may not necessarily fit the data well. In order to address this problem, or at least partially, linguistic hedges can be employed.

The concept of hedges has been proposed quite early-on in fuzzy set research and were introduced in [33]. A linguistic hedge produces a new fuzzy set by changing the original fuzzy set, in a predefined and interpretable manner. The interpretation of the resultant set emanates from the original fuzzy set and a specific transformation that the hedge suggests. The original fuzzy sets are not changed, but the hedged fuzzy sets provide modifiable means of modelling a given problem, and therefore more freedom in representing knowledge in the domain.

In previous research genetic algorithms have been applied to obtaining compact sets of linguistically hedged fuzzy rules from data, by translating precise models [20] into linguistic ones. In this work a classifier system (XCS [31]) will be used for fuzzy rule extraction. XCS is an extension of the Michigan style classifier systems [10] that has some very interesting features. A standard Michigan style classifier system is based on a single rule set, with each rule represented in the form; condition→action. The learning mechanism works by adjusting certain values associated with each rule based on the feedback given by the environment. XCS extends these classifier systems to provide complete maps (which can produce an estimation of reward for each of the possible outputs with respect to any given input). The classifiers (i.e. rules) will be of the form; condition+action→reward. In addition, XCS includes a method, called covering, which ensures that each possible input has a rule with a matching condition for each available action. These two factors mean that such an XCS classifier system will always provide, for any input, an estimation of rewards to be obtained for choosing and executing any of the outputs. This provides additional insight into the relationships between the input data and all the possible classes, thereby offering a better understanding of the underlying problem which is being modelled.

This paper demonstrates the results of adapting XCS to create linguistically hedged fuzzy classifier systems. The fuzzy XCS with linguistic hedges, denoted as LF-XCS, is tested on a real world problem domain; identification of glass type (a forensic dataset obtained from the Forensic Research Institute, Krakow, Poland). The results of this experimental study are also discussed.

The rest of the paper is organized as follows. Section II summarises the two main approaches to fuzzy modeling - precise and linguistic. Next, section III presents the basic XCS classifier learning system and the modifications proposed to transform it into LF-XCS. In section IV the dataset for forensic glass identification is examined, including a brief description of how the data was obtained. Section V shows the experiments performed and the parameters used. The results of the experiments are analysed in section VI. The paper is concluded in section VII, with a short discussion of future work.

II. Precise and Linguistic Fuzzy Modeling

In the first generation of fuzzy systems, inference rules were provided by human experts. The domains of the input variables were subjectively partitioned using fuzzy sets. The definitions of such fuzzy sets - to be used later in the rules - were also generated by experts. The domain partitions were required to satisfy certain properties, which made these early systems completely transparent. These properties included distinguishability, completeness, etc. [29]. Such an initial approach quickly developed into fuzzy grid partitions and was often used in implementing logic controllers.

One of the main disadvantages of this early approach was that the knowledge acquisition process of obtaining the required expertise became very difficult. This of course has led to a bottleneck in the development of such systems. Furthermore, not all knowledge obtained from experts was optimal, accurate or even necessarily consistent. However, in the digital era, the amount of data available about typical applications has been growing considerably in many problem domains. A logical step was to apply machine learning algorithms to automatically optimise existing systems, or even to create them from scratch using the data. Most of these algorithms were based on supervised learning techniques [1, 2, 3, 13, 27, 30].

Many of these learning systems (mostly for logic controllers) used fuzzy grids. This had the advantage of full covering of the input space and offered a simple table-like way of providing the rule base. Yet this approach can only be used when the dimensionality of the input space is small. As dimensionality increases the total number of possible rules explodes exponentially, rendering the system impractical.

Another problem associated with such learning systems is that the available data usually clusters only in certain areas, with most of the input space empty. In these sparsely populated systems the use of a full grid is a waste of resources, where the full grid is the cartesian product of all fuzzy sets defined on each input variable for each possible output. This is because many of the rules will cover empty areas of the input space. Therefore, an initial potential optimisation would be to remove these empty-covering

rules. This optimisation would assist in obtaining a (hopefully small) subset of the grid cells. These possible outputs are either a singleton fuzzy set defined in the domain of the output variable, or a compound value from these fuzzy sets. A very basic algorithm would enumerate all of these rules and select those that fulfill some (possibly error based) criteria as in [16]. Unfortunately the potential number of rules in high dimensional problems again makes this approach impractical. Other algorithms, e.g. the widely known Wang and Mendel method [30], would instead use the available input examples to locate useful rules. With strong partitions (namely, each example can only belong to, at most, two different sets and the sum of all membership values is equal to one) the maximum number of rules that cover one example is 2^L with L being the number of input variables. This partially alleviates the problem of testing the $\prod_{i=1}^{L} D_i$ different rules (with D_i being the number of fuzzy sets defined in the input variable i) of any exhaustive methods, at least for moderately-sized problem domains.

Other methods have been proposed to deal with situations where the potential number of rules made exhaustive enumeration difficult. In particular, evolutionary techniques have been the most successful [9]. However, such fuzzy systems raise the problem of coverage. Undefined or undecidable areas have been dealt with using default values or using fuzzy sets with infinite tails (such as sigmoid or gaussian). Here, the undecidability is caused by the fact that certain algorithms may accept rules only when the error over the training set is below a prescribed threshold. A simple case is one potential rule covering two incompatible examples. The learning system may decide to correctly classify just one (while incorrectly classifying the other), to average both outputs, or to declare the zone undecidable and hence not to include the rule in the learned system.

Approaches which employ fuzzy grids showed that severe limitations were placed upon the granularity of the input space. In light of this, optimisation algorithms were introduced which allowed a relaxation, such that the subjectively defined fuzzy sets were allowed to be fine tuned. In particular, expert or user predefined fuzzy sets were allowed to change in order to best fit the data. Unfortunately this often severely disrupts the linguistic interpretation, - even destroying it in many cases. Following this approach, not only are the sets modified, but also the original grid may be discarded, while creating fuzzy sets for the exclusive use of possibly just one single rule. That is, each rule would have its own fuzzy sets which are not shared with other rules, known as scattered fuzzy partitions [11]. This forms what is termed precise (aka. approximative) fuzzy modeling. The resulting systems sacrifice interpretation for accuracy.

There are alternative proposals that do not follow the approximative approach. Such work, following the original linguistic (or descriptive) fuzzy modelling, is mostly concerned with the development of fuzzy systems where the fuzzy set definitions are human-defined and not allowed to be modified. Of course, this gives rise to the issue of granularity, regarding the actual fuzzy partition of the problem domain, which must be taken into consideration. Given that humans seem to be able to handle with ease only around 7 ± 2 different aspects for a concept [21], the cardinality of each domain partition should remain close to this figure. This means that the number of descriptive fuzzy sets available for use in the rules is rather small. Furthermore, as mentioned previously the definitions of the fuzzy sets are generated by the human user and can not be modified to better fit the training data (because it may otherwise destroy its interpretation). This imposes a *de facto*, fixed grid in the input space.

It is highly unlikely that the data will fit in a convenient manner in such a crudely fixed grid. In fact, great rooms usually exist for a given grid to be changed to accommodate the data. This is precisely what would have led to the precise modelling approach in the first place. So, is there a mechanism that can provide finer or modified granularity of the grid while retaining the linguistic fuzzy sets exactly as defined by humans? One way in which this can be achieved, whilst simultaneously allowing for better precision is to maintain the set definitions, and to employ linguistic hedges [33]. A linguistic hedge produces a new fuzzy set by altering (in a predefined and interpretable manner), another fuzzy set. The interpretation of the resultant set emanates from the original fuzzy set and the specific transformation that the hedge suggests. The original fuzzy sets remain unaltered, but the hedged fuzzy set provides another option to the system, and therefore more flexibility in representing the knowledge of the domain.

For simplicity, and ease to use, linear piece-wise fuzzy sets have been used extensively. This work also adopts such fuzzy sets. In particular, this research makes use of trapezoidal and shouldered sets. Note that the definition of traditional hedges does not produce substantial changes in these types of fuzzy sets. To address this issue, a new definition of hedges has been proposed in [19, 20]. This new approach produces better results when applied to linear piece-wise sets. This paper uses the exact same definition of such hedges in implementation. Therefore, detailed hedge definitions are omitted herein but can be found in [19, 20].

III. XCS Learning Classifier System

As stated previously, Michigan style classifier systems [10] are based on a single rule set in the form of condition→action. The learning mechanism works by adjusting certain variable values associated with each rule, based on feedback about the action errors, as well as discovering new and better rules. Discovery of new rules is obtained by mating or by mutating old rules in a manner employed by Genetic Algorithms (GAs). Rule conditions usually include a "don´t care" state in depicting many variables, allowing for generality.

The XCS learning Classifier System [31] is an extended clas-

sifier system. A distinct feature is that it produces a complete map ($X \times A \longrightarrow P$ with X being the input space, A the action, or classification if the task is such, and P the prediction space about the rewards). This map extends the dimensionality of a given problem, including both the input variables and the output variables (the actions). What XCS produces is a prediction of expected rewards for each action. Normal classifier systems [12] usually use a map $X \longrightarrow A$ that maps inputs onto actions. This difference adds extra complexity to the problem, but it adheres more closely to the philosophy of reinforcement learning (that is the base of its learning mechanism) [24, 28]. Such learning aims to obtain estimations of the consequences of every possible action to be performed in a given situation.

XCS has mechanisms to promote generality in the learned classifiers. It has a preference for having many "don't cares" in the condition part; such variables are not considered in the firing of the rule. XCS can learn non-sequential tasks, including classification, which is the focus of this paper. However, it can also learn sequential tasks, that is, sequences of actions to obtain rewards. A brief outline of how XCS works is given below. For a detailed treatment of XCS and discussions of its features see [15]. A description of the standard algorithm is also available in [6].

A. XCS working procedure

The basic procedure of XCS is as follows. The current object is matched with the patterns of the condition part of the classifiers, with each such classifier forming a classification rule. Classifiers that are matched constitute the *match set* and are grouped into subsets that each share the very same action (output). Classifiers in the same subset combine their predictions weighted by each classifier's precision and, in the case of linguistic fuzzy-XCS, its degree of matching (see equation 1 in section III-E). So, an estimation of the reward is obtained for each possible action, forming what is known as the *prediction matrix*. If a particular action has no matched classifier a new classifier is created that matches the example and advocates for that action (this is called "covering") so that every action has always an estimation of reward.

Note that in reinforcement learning the training phase is carried out by alternating between two modes of selection for the next action to perform. These two modes are exploration, and exploitation. In exploitation the system checks the most promising action in order to continue the assessment of the most likely action to be used in the future. In exploration, it selects a random action in order to check other alternatives. Both modes are necessary in order to perform training correctly, as learning is only carried out on the executed actions, hence all actions should have potential participation, not only the most promising ones. When XCS is in exploration mode, a random action is selected, whilst in exploitation mode the action with highest predicted reward is chosen.

The subset of matched classifiers that have advocated the chosen action is called the *action set*. The action is then carried out (i.e. the class is decided for a classification problem) and a reward of the effect of performing such an action (or the success or failure in classification) is obtained. This reward is fed back to updating the values of the prediction, fitness and error of the classifiers within the *action set* itself.

B. Rule discovery in XCS

New rules are discovered in XCS by: (1) ensuring coverage and (2) exploiting a GA.

1) Covering method

As indicated previously, the covering mechanism works by following the rule that given an example, if there is an action that has no matched classifiers then a new classifier is created. The condition part of this new classifier has to match the current example. In particular, certain parts of the condition will have a given probability $P_{\#}$ of being "don't care", and the rest are to be chosen in a way that matches the current input. The output of the classifier is the uncovered action. This new classifier is included in the current population of the GA (that is to be briefly introduced next) and in the match set, providing a prediction for the previously uncovered class (although being random). If more than one action is not covered the "covering" procedure is repeated for each. Note that this is just one of the main mechanisms to obtain complete maps. Quite often XCS starts with an empty population that grows with the examination of the training examples until the employed GA is able to activate and support the creation of new classifiers.

2) GA exploitation

XCS uses a niche GA, that is, only a group of classifiers are candidates for being parents. Earlier versions of XCS [31] used the match set as the parent niche, but later the action set was employed. The GA is activated using a frequency method. If the average time since the last activation of the GA for the classifiers in the action set is greater than a given threshold θ_{GA}, then the GA is started.

There are two genetic operators in XCS. A standard two-point crossover with probability P_{χ}, and mutation. The mutation is special in the sense that any offspring that undercarries mutation has yet to match the current input. It works by changing each condition with probability P_{μ}. Mutation can be applied to the action part also. Of course, there is no need to enforce matching in the crossover as any crosses of two matching conditions are matching conditions.

C. Removal of rules

XCS conducts an optimisation procedure called numerosity. It allows several copies of the same classifier to remain in the population, with the same values for prediction and fitness. These are called "macro-classifiers". When these classifiers

are activated their effects are weighted using this numerosity figure. Therefore, an XCS system usually has an actual number of classifiers (the total of those classifiers that are really different from one another) and an operative number of classifiers (the sum of all classifiers).
XCS has a pre-defined maximum population size (including the numerosity of macro-classifiers). If this number is exceeded then some rules will be discarded to make room for new rules. Selection for deletion is performed over the whole population (note that, for breeding, selection is made over the action set only). The likelihood of a rule being discarded relies on two factors. The first is the average size of the action set of a classifier. This is intuitive, in the sense that, if a rule usually fires along with many other rules (that have the same output) then there is a possibility that the rule is not required. The other rules would cover the same examples. The other factor is classifier fitness, that is, its quality. If the fitness is less than (a fraction δ of) the average fitness of the population, the chance of being deleted grows substantially. Note that, as rules need time to accurately assess their true fitness younger rules (i.e. those with low experience) do not apply this latter factor. Clearly, if the numerosity of a certain macro-classifier reaches zero then this classifier is completely removed from the system.

D. Promotion of generality in XCS

Generality is achieved in XCS through several mechanisms. Two particular methods are the *GA Subsumption* and the *Action Set Subsumption*. Subsumption is a mechanism which replaces classifiers with more general versions. GA subsumption is activated when the GA generates new offspring. If the new classifier is completely covered by the parent (that is, the parent is more general), and this parent has a high precision and a minimum experience, then the child classifier is discarded and the numerosity of the parent increased by one. Action set subsumption happens for every GA cycle. In each action set the most general classifier is found (usually the one with most "don't care" states). If such a general classifier has a high precision and is experienced (namely, its precision estimation is reliable) then all classifiers in the action set that are completely covered by the most general are deleted. For each classifier deleted the numerosity of the most general classifier is increased by one. This latter subsumption may be activated more often than GA subsumption and thus may be used more often. However, it is worth remembering that this will only happen if an experienced general classifier with a high fitness exists in the niche.
There are other mechanisms that also promote generality, albeit in a more indirect fashion, when compared with the previous methods. One is to apply mutation pressure. Mutation can be biased to generate more "don't care"states in the condition part, thereby creating more general classifiers than simple random mutation [5].
Another mechanism is the one known as set pressure, as identified in the so-called generalisation hypothesis [31], which indicates that XCS has a natural tendency to evolve accurate and maximally general classifiers. This set pressure arises as a consequence of the aforementioned fact that the GA selects parents in the action set only, but deletion of classifiers is applied to the whole population. The rationale for this is as follows: if a rule is quite general it will fire more often than those that are less general. Firing a rule more often also means that the rule will appear in the action set more often than rules with a more specific antecedent. Of course, if it appears more often in the action set this means that they will act more often as parents to reproduce new offspring. Therefore the parents of the new classifiers tend to be more general than the average classifier in the population. Offspring of more general parents tend to be more general than the average classifier in the population too. As a result, the offspring created by the GA tend to be more general than the average classifier. However, deletion takes place on the whole population so the classifier to be replaced is usually less general than the new classifier. Thus, there is a pressure to replace those less general classifiers with more general classifiers.
There are other factors involved in the set pressure but this factor provides substantial pressure toward general rules. A theoretical formulation and empirical test of generalisation issues in XCS can be found in [5].

E. From XCS to a linguistic Fuzzy XCS

Several changes have been introduced to adapt XCS for a linguistically hedged fuzzy environment. While extending the XCS classifiers to represent linguistically hedged fuzzy rules the resulting learning mechanism is expected to inherit the underlying approach that traditional XCS algorithm possesses. Indeed, the basic XCS algorithm remains unchanged in this work. Only the interpretation of some of its components is extended to entail the use of fuzzy representation. For instance, the concept of matching a current observation with the classifiers is changed to allow for a partial match. Accordingly, the mathematical expressions employed to calculate predictions are extended by the introduction of degrees of matching. Also, the concept of subsumption is reframed as full inclusion of the fuzzy spaces defined by the rule antecedents. However, these changes do not affect the learning procedure of the XCS at all, they simply modify parts of the computational details. Such modifications are briefly summarised as follows.

1) Representation

Classic classifier systems were designed for boolean environments. The original alphabet was ternary (the third state being the "don't care"). This has been extended to real [26, 32], or fuzzy [4, 7] environments, including the proposal for variable alphabets for different variables [17, 18]. These extensions allow for different types of data to be represented covering variables that are qualitative, quantitative (discrete and continuous) as well as linguistically fuzzified (with hedges).

Extension for qualitative or nominal data, which involves no order relations amongst different values, is performed in a similar way as in boolean environments. The alphabet for such variables is the qualitative terms of the given domain plus a "don't care" symbol. An illustrative example of this type of data can be found in [17]. There, the KDD dataset includes a variable that is the protocol used for internet connection. The alphabet for that variable included "don't care", http, udp, tcp, etc.

In quantitative data non fuzzified non-sorted intervals are used. Note that intervals are regarded to be "sorted" if the pair of values $[a, b]$ that define an interval are set such that $a < b$. In the present work, non-sorted representation is used as it avoids some biases, as argued by [26].

Fuzzified variables also include a "don't care" value. When used in a rule, such a value is equivalent to state that the corresponding variable may be of any value of the domain of that variable or a missing value. This implies that such a rule antecedent is always completely true whatever the variable may be. As rule antecedents used in LF-XCS are conjunctively linked (that is, they only use the AND connective), these "don't care" values can be safely removed from the antecedent expression owing to the boundary property $T(x, 1) = T(1, x) = x$ of any T-Norm that may be used to implement the AND operator [23]. Therefore, rules are built using only antecedents whose values are different from "don't care".

Note that the number of hedges allowed is variable but more than two per fuzzy set severely reduces the interpretability. For this reason the maximum number of hedges is two (though this may be retrieved if needed). Reduced numbers of hedges reduces the degree of freedom of the model, making it harder to fit the data, of course.

2) Prediction and learning updates

For classification tasks the output is a crisp value, namely the identified class. In classical XCS the evidence for each class is obtained by aggregating the reward prediction of matched classifiers of that class weighted by its fitness. The class with the highest expected reward is the winner. With fuzzy classifiers a partial (fuzzy) match is then possible. The contribution of those classifiers that partially match is therefore weighted by the degree of the corresponding matching as well.

The original expression of XCS for obtaining the reward prediction is shown in equation 1, with $P(a_i)$ denoting the predicted reward when action a_i is taken (or class a_i is chosen when addressing classification problems), F_c is the fitness of classifier c (it has to be in the match set for that class, that is, $c \in [M]_{a_i}$), and p_c is the reward prediction of that classifier.

The modified expression for the proposed linguistically hedged fuzzy XCS is shown in equation 2. In this equation S_c is the matching strength of the condition. The matching (or firing) strength is calculated as a T-norm (in the experiments that follow it was implemented with the minimum operator) of the membership values. The membership values are those that each input variable (of the current data item) has with regard to the hedged fuzzy set condition (for the respective variable) of the classifier. The mathematical expression of the matching strength can be seen in equation 3, where $\mu^c_{H_i}(x_i)$ is the membership value of the ith component of data item x with respect to the hedged fuzzy set H_i, that is the ith condition of the classifier c.

$$P(a_i) = \frac{\sum_{c \in [M]_{a_i}} F_c \cdot p_c}{\sum_{c \in [M]_{a_i}} F_c} \tag{1}$$

$$P(a_i) = \frac{\sum_{c \in [M]_{a_i}} F_c \cdot p_c \cdot S_c}{\sum_{c \in [M]_{a_i}} F_c \cdot S_c} \tag{2}$$

$$S_c = T_i(\mu^c_{H_i}(x_i)) \tag{3}$$

During the learning process of XCS the updating expressions for both fitness and reward prediction are modified in a very similar way and are therefore omitted here. They also use the firing strength of the classifier as an extra weighting factor.

3) Subsumption

The subsumption mechanism is essentially the same as in XCS without linguistic hedges. Subsumption means inclusion, i.e., the input space covered by the subsumed classifier is a subset of the input space covered by the subsumer. In practical terms a hedged fuzzy set is another fuzzy set modified from the original set or the union of several fuzzy sets, including the original. Therefore, given two hedged fuzzy sets, namely H_a and H_b, it can be considered that H_a is subsummed in H_b if the former is included in the latter. That is, if $H_a \bigcap H_b = H_a$.

4) Mutation operator

Mutation has been improved to allow it to modify hedges and fuzzy sets, or to change the whole condition to "don't care". Once a condition has been selected for mutation (with probability P_μ) the procedure works as follows. With probability $P_\#$ it changes the whole condition to "do not care". Otherwise, with $P_{\mu add}$ chance it adds a new hedge (or change one if the maximum number of hedges allowed is reached), with $P_{\mu del}$ chance it removes a hedge (or add a new one if no hedge is present), and with $P_{\mu set}$ it changes the base set. After such changes are implemented, it is checked if the corresponding input variable value is still covered by the mutated hedged fuzzy set. If this is not the case the mutation is undone and the process repeated. If, after several tries, no mutated set is found that matches the input variable, then the mutation is discarded in that variable condition. Note that this check must be done as the mutated classifier has to match the current input condition.

Apart from the previously mentioned changes the XCS procedure is as described in [6]. During the training phase XCS usually uses a 50/50 exploration vs. exploitation scheme. It

is at this time that the covering mechanism must be activated. However, in order to test an emerging model this mechanism must be disabled. It is important to point this out since some work in the area does not disable covering while testing. If enabled, and if a particular test data item is not covered by all possible actions, the covering mechanism would create new classifiers (and possibly delete some to make space) thereby completely altering system behaviour.
Note that disabling covering may produce "unknown" classifications. It is a matter of taste if such an "unknown" class is produced whenever a single action is not covered (but still having evidence in favour of other actions) or when all actions are not covered (no evidence for any action whatsoever). In this work the former approach is taken.

IV. An Application Problem

This section introduces the domain problem. The data is a forensic dataset, obtained from the Forensic Research Institute, Krakow, Poland. It contains information about various types of glass and their chemical and physical properties. The samples were obtained through the following procedure: One large piece of glass from each of 200 glass objects was selected. Each of these 200 pieces was wrapped in a sheet of grey paper and further fragmented. The fragments from each piece were placed in a plastic Petri dish. Four glass fragments, of linear dimension less than 0.5mm with surfaces as smooth and flat as possible, were selected for examination with the use of an SMXX Carl Zeiss (Jena, Germany) optical microscope (magnification $100\times$).
The four selected glass fragments were placed on self-adhesive carbon tabs on an aluminium stub and then carbon coated using an SCD sputter (Bal-Tech, Switzerland). The prepared stub was mounted in the sample chamber of a scanning electron microscope. Analysis of the elemental content of each glass fragment was carried out using a scanning electron microscope (JSM-5800 Jeol, Japan), with an energy dispersive X-ray spectrometer (Link ISIS 300, Oxford Instruments Ltd., United Kingdom).
Three replicate measurements were taken from different areas on each of the four fragments, making twelve measurements from each glass object, but only four independent measurements. The four means of the measurements were used for the analysis. The measurement conditions were accelerating voltages 20kV, life time $50s$, magnification 1000 - $2000\times$, and the calibration element was cobalt. The SEMQuant option (part of the software LINK ISIS, Oxford Instruments Ltd, United Kingdom) was used in the process of determining the percentage of particular elements in a fragment. The option applied a ZAF correction procedure, which takes into account corrections for the effects of difference in the atomic number (Z), absorption (A) and X-ray fluorescence (F).
The selected analytical conditions allowed the determination of all elements except lithium (Li) and boron (B). However, only the concentrations of oxygen (O), sodium (Na), magnesium (Mg), aluminium (Al), silicon (Si), potassium (K), calcium (Ca) and iron (Fe) are considered further in this work as glass is essentially a silicon oxide with sodium and/or calcium added to create a commonly produced glass, and potassium, magnesium, aluminium and iron added to stabilise its structure and modify its physio-chemical properties. Histograms of the distributions of the data can be found in figure 1.

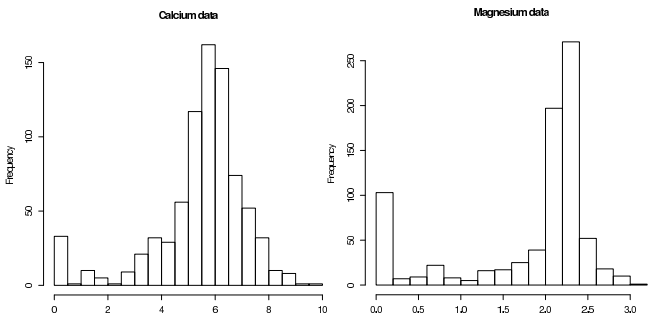

Figure. 1: Data distributions for Calcium (left) and Magnesium (right).

Table IV presents the data distribution by classes. It reflects the average number of examples in the training and testing folds. It also shows a severe class imbalance in the number of examples of each class. Some classes, such as car or building windows glass are much more represented than optical glass or glass containers. Note that this usually represents a significant challenge for most learning algorithms.

Table 1: Data distribution by instances

class	avg. training	avg testing	total	% of total
bulb	93.6	10.4	104	13 %
car wind.	219.6	24.4	244	30.5 %
headlamp	57.6	6.4	64	8 %
optic	21.6	2.4	24	3 %
containers	25.2	2.8	28	3.5 %
building wind.	302.4	33.6	336	42 %
total	720	80	800	100%

A. Data preparation

LF-XCS requires fuzzy sets to be defined for each of the input variables. In order to properly claim the approach is indeed transparent and purely linguistic such fuzzy sets should not be engineered to fit the data. That is, no information measure extracted from the data guides the fuzzy set definitions. For example, histograms shown above are included to illustrate the properties of the data but are not used in any way to define the fuzzy sets. The fuzzification was carried out proportionately with respect to the size of the universe of discourse of the individual variables. The distance between its maximum and minimum value within the data set is divided such that all fuzzy sets approximately cover an equal

range of the underlying real values, with soft boundaries of course. Generic labels namely, *tiny*, *small*, *medium*, *large*, and *huge* were attached to the sets in ascending order. Note that this does not neccessarily correspond to what an expert would use, in fact each expert/user would define their own fuzzy definitions and labels accordingly. This fuzzificatation is used herein assuming that no expert-given labels are available. In real applications, the system will learn using the experts/user own words, rather than vice-versa.
All attributes are real-valued, that is, no discretization is performed. The number of descriptive fuzzy sets available for the system to use in the rules is relatively small. This is because, as already mentioned, the number of different concepts that a human seems to be able to handle is around 7 ± 2 [21]. Empirically, it is normally the case that for five labelling terms, experts can easily find different words while, if asking for more, they tend to struggle to reach sensible words that they are content with.

V. Experiments

The principal aim of this research is to demonstrate that the effects of using hedges result in improved accuracy of the classification system, whilst simultaneously maintaining the interpretability of the XCS models. Two main experiments are therefore performed using the same learning algorithm, LF-XCS, including the same algorithmic parameters (see section V-B below). Two separate experiments are performed; in one experiment no hedges are allowed, while in the other up to two hedges may be used in learning the classifier conditions.

A. Experimental considerations

XCS generates maps that are complete. There are rules that suggest that an area of the input space belongs to a given class (classifiers that produce outputs with a high reward). Yet, at the same time, there are rules, covering parts of that same area, suggesting that the area is not of the other classes (classifiers that produce outputs with a low reward). Therefore it is not unusual to find rules with antecedents covering same areas but with different consequents. One of the rules may have a high reward and the rest a low reward. The former suggests the class of that area, with the latter confirming that such an area is not of the remaining classes. All these rules will have a high fitness (that is, of course, if the model is sound). This is quite different from most rule-based systems. Typical rule-based systems contain only one rule that suggests a particular classification and do not include other rules that would point to a "not this class" result. This means that XCS has more rules than systems built using traditional methods. It could be argued that this increase in the number of rules reduces transparency. However, it provides a more informed decision, as it shows the evidence for each possible outcome and not just for the overall winning result.
In order to assess how many rules XCS might have represented in a traditional form, where only rules concerning the type "is class X" (i.e. confirming a certain classification) are involved, the rules of a low reward are removed. This allows for the computation of classification accuracy in using just such rules. Although this assessment is carried out in a fairly crude way, it should allow the reader to have an idea of the resulting ruleset complexity, in terms of the number of rules, in comparison to systems that are possible alternative to the present approach.
It is worth reiterating that Michigan style systems [10] include a mix of well tested and high precision rules in the population *and* also new and recently created candidate rules that are still being evaluated. However, the Pittsburgh approach [25], which is also quite popular, works in a rather different way. In Pittsburgh style systems the members of a population are independent sets of rules. All the rules in such individual members are tested as a block, with a single fitness for all. These members are not modified afterward – they are either retained in the population as potential solutions (or potential parents of better solutions) or discarded. In so doing, Pittsburgh style systems usually produce very compact rule sets. This is because fitness tends to be badly influenced if all rules, or their synergic combination do not perform well. That is, only performance-wise useful rules are included in the sets. Many induction algorithms use the Pittsburgh approach so it is difficult to compare, in terms of number of rules, with systems which follow the Michigan approach. In order to provide an idea of the number of rules that are mainly responsible for the performance of LF-XCS (Michigan approach), an evaluation of the results after a simple removal of rules with low fitness is also presented.
Note that these modification procedures are expected to degrade the system performance, as XCS is designed to include all of these removed rules, especially the "not this class" type classifiers. However, such rule removal will produce a figure which is easier to compare over the number of rules. The particular thresholds used in this work will be 10% of the maximum fitness allowed (if it is falls below such a value the classifier is removed) and 10% of the maximum reward. The former removes the new, insufficiently tested rules while the latter deletes the "not this class" classifiers. These thresholds seem reasonable in order to provide an acceptable estimation of the corresponding number of rules in a non-complete and Pittsburgh type of approach.
The method used to calculate the performance of the generated system is a 10-fold cross validation [8] with 20 runs for each fold. Each execution of XCS (one per fold of each run) has 25,000 training cycles C. Half are exploring cycles, and half exploiting. In non sequential tasks, such as classification, the exploiting cycles serve to evaluate the learning progress.

Table 2: XCS parameters.

$N = 1000$	$P_{\#} = 0.5$	$P_{\chi} = 0.8$	$C = 25000$
$P_{\mu} = 0.04$	$P_{\mu add} = 0.2$	$P_{\mu del} = 0.2$	$P_{\mu set} = 0.1$
$\alpha = 0.1$	$\beta = 0.2$	$\delta = 0.1$	$\nu = 5$
$\theta_{GA} = 25$	$\theta_{del} = 20$	$\theta_{sub} = 200$	$\epsilon_0 = 10$

B. XCS parameters

There are a moderate number of parameters which must be defined for the XCS. Most of them are set to values that have been suggested in the literature and not fine tuned for the present specific problem. In order to avoid an excessive pressure to generalisation, and as suggested in [22], the parameter θ_{sub} (experience of a classifier to be considered as a subsumer) is set to 200. An extensive discussion of all parameters can be found in [15]. There are also a few additional parameters added for the linguistic fuzzy extension. The final values used in this work are listed in table 2.
Finally, the reward value for a correct class given to the classifiers in the action set is 1000. For an incorrect class the reward values is 0. These are typical values from the XCS literature for classification problems.

VI. Results

Experimental results for the given dataset are presented in table 3. The *Trn* column shows the average error percentage over all the experiments for the training folds, with the confidence intervals at 95% (which are calculated as $\pm T_{Val95\%}(n) \cdot \sigma/\sqrt{n}$, with $n = 20$, $T_{Val95\%}(20) = 2.09$ and σ being the standard deviation). The *Tst* column presents the average error percentages for the testing folds. The next column, *# R* gives the average numbers of macro-classifiers (different rules) of the final LF-XCS systems. Finally, the last column labeled as *Rsz*, lists the average number of the conditions in the antecedent part of the rules that are different from "don't care"; such a number is also known as the size of the rule concerned.

A. Observations and discussions

The results associated with the row of LF-XCS in table 3 show the different values of the learned system prior to any rule pruning. There is a 4% increase in accuracy when hedges are used. This result is statistically significant. When the rules with low fitness are removed there is a drop of about 1% accuracy, whilst reducing the number of rules in the system by more than 200. Interestingly the remaining rules are, on average, slightly more general, as shown by a 0.2 decrease in average rule size. This rule size reduction is reverted when the rules with low rewards are also removed. The final number of rules, when most rules retained are those that only advocate for a particular classification, is about 50. These results show that when the rules with low reward are removed, not only are the number of rules obviously reduced, but the average generality of rules also degrades significantly.

With regard to rule reduction a first, albeit incorrect, assumption may be the following: As the map is complete, if the problem being modelled has N different classes, then for each rule advocating a particular class there might be about $N - 1$ rules indicating that that rule does not lead to any of the remaining classes. So given the current problem has 6 classes the removal of the "not class X" rules should produce a reduction, in the number of rules, around the ratio 5 to 1. Yet the experiment results show a reduction of barely 2.5 to 1.
The assumption of $N-1$ to 1 reduction in the number of rules would be correct if both types of rules (high and low reward), would be equally general. However, as clearly demonstrated by this experimental evaluation, the rules that advocate for low reward (that is, for "not class X") are more general than the average rule. This can be expected since a quite general rule can cover wide areas where there may be examples of several different classes as as these classes are not that particular class "X". Nevertheless, rules of type "is class Y" must cover specific areas where only class "Y" is present, and these rules tend, on average, to be much more specific. If the "not class X" type rules are removed, the remaining ones are, as demonstrated by the results, more specific. These effects occur for both hedged and non-hedged experiments, in similar proportions. Note that while the number of conditions is an indirect way to measure specifity (even more so with the use of hedges), it correlates, in general, with the size of the area that the antecedent condition covers.
As $P_{\#}$ is 0.5, and the number of variables of the problem is 8, the average size of the new rules created by XCS is therefore 4. However, the average rule size returned by LF-XCS is of 3.29 conditions per rule. Thus, the LF-XCS learning procedure is capable of selecting rules that are, on average, more general. This reduction in size of the average classifier at the end of the learning process shows the the effects of the pressure upon generality. Rules with few conditions are easier to interpret so this result shows a very positive outcome of utilising LF-XCS.

B. Analysis and comparison

Table 4 shows the confusion matrix of the average test results. Each cell shows the average number of examples that the system classifies as per the column while actually being examples of the class as per the row (see table IV for the average number of examples for each class in the testing folds). The last column indicates the percentage of correctly classified instances for each class. The last row shows the percentage, grouped by the class that the system predicted, which is indeed of the correct class. For example, 96.5 % of the bulb glass instances are correctly classified, while 93.7% of the instances that the system classified as bulb glass are indeed bulb glass examples. There should be another column labelled "Unknown" but in both experiments this contains zeros only and hence is removed for clarity. Confusion matrices for XCS after removal of low reward and low fitness

Table 3: Results for Polish Glass Dataset

Experiment	Trn	Tst	# R	Rsz
	LF-XCS Using Hedges			
LF-XCS	17.09 ± 0.36	19.95 ± 0.48	351.44 ± 1.60	3.29 ± 0.03
Removed Low Fitness	18.52 ± 0.42	21.55 ± 0.55	125.08 ± 0.90	3.08 ± 0.02
Removed Low Fitness and Low Reward	20.70 ± 0.50	23.91 ± 0.51	52.5 ± 1.00	3.75 ± 0.03
	LF-XCS Not Using Hedges			
	Trn	Tst	# R	Rsz
LF-XCS	22.15 ± 0.41	23.95 ± 0.46	347.58 ± 1.06	3.05 ± 0.02
Removed Low Fitness	23.03 ± 0.45	25.17 ± 0.57	130.83 ± 1.26	2.84 ± 0.02
Removed Low Fitness and Low Reward	25.23 ± 0.44	27.64 ± 0.57	50.26 ± 0.43	3.68 ± 0.02

rules have "Unknown" columns with values different from zero. Such results are considered as missclassification and counted as errors.

It is interesting to note that the bulb, headlamp and glass containers classes are correctly discriminated. Optical glass is a very specific class, the chemical composition is substantially different from the other classes and it is therefore easily classified. However, classes representing car and building window glass are usually confused about 25% of the time. This happens because of the similarities in the float glass manufacturing process. The methods by which building window and car window glass is created are basically the same (float glass).

A study of applying different fine-tuned, precise approach-based classification techniques to the present problem can be found in [35]. Although such work uses a glass dataset which is different from the one that is used in this research, the examples are obtained using the same technique as described in section IV. In that study car and window glass are considered in many experiments as a single class and experiments are carried out to distinguish between just these 2 classes (car window and building window). This gives an accuracy of a little over 80%, not far from the 75% of LF-XCS. This difference in performance becomes even less significant when one considers that LF-XCS is a linguistic approach and that the 25% misclassified examples for car and building also includes confusions (albeit small values) with the other classes. In particular, it is important to note that in the existing results of [35], the confusion between building and car window glass, the training was performed over just these 2 classes, and not for all 6 classes as in the current research.

Interestingly, the heavy class imbalance does not prevent LF-XCS from properly classifying the minority classes. These results conform to the observation in that XCS can be particularly resistant to the class imbalance problem, as revealed by the research reported in [22].

Looking at the confusion matrix again, when no hedges are allowed (table 5), the classification accuracy over the bulb, optical and glass containers classes are maintained. This seems to reveal the fact that most of these classes are separable from each other with outliers falling into grid areas of other classes. Outside the core of these classes, with a more restrictive grid, there are certain boundary areas that can not be as precisely delimited. Some examples of building windows fall into the the fixed cells that are classified as headlamp or glass containers. Similarly certain examples of headlamp are classified as glass containers or car windows. However, given a fixed grid, it is harder for the non-hedge approach to discriminate properly between car and building window glass which are more mixed, usually by misclassifying more car window glass as building glass.

```
IF O is Medium AND Mg is Upper Small
   AND Na is Medium AND Si is Below Large
      THEN Class Headlamp
      REWARD 999.9
      NUMEROSITY 10

IF Na is Tiny
      THEN Class Bulb
      REWARD 0
      NUMEROSITY 16

IF Mg is Lower Medium AND Si is Large
  AND Ca is Very Large AND Fe is Tiny
      THEN Class Glass Containers
      REWARD 999.9
      NUMEROSITY 17
```

Figure. 2: Three example classifiers

C. *Example of classifiers*

In order to illustrate the ability of LF-XCS to retain model interpretability figure VI-C shows the first three classifiers of one run. Note that the second classifier has a reward of 0, meaning that it is a "not this class" type of classifier that means that if the content of Sodium is tiny then the sample can be discarded as being glass from a bulb. The other classifiers are self explanatory as they are pure linguistic expressions.

Table 4: Confusion Matrix for XCS using hedges (average for testing folds)

	Predicted						
Actual	bulb	car wind.	headlamp	optic	containers	building wind.	Actual Correct %
bulb	10.04 ± 0.10	0.05 ± 0.03	0.19 ± 0.06	0.00 ± 0.00	0.03 ± 0.02	0.09 ± 0.04	96.58 ± 0.78
car w.	0.05 ± 0.03	18.15 ± 0.35	0.24 ± 0.06	0.00 ± 0.00	0.12 ± 0.05	5.84 ± 0.36	74.42 ± 1.46
headlamp	0.36 ± 0.09	0.07 ± 0.04	5.80 ± 0.12	0.01 ± 0.01	0.06 ± 0.03	0.10 ± 0.05	90.59 ± 1.62
optic	0.00 ± 0.00	0.00 ± 0.00	0.01 ± 0.01	2.36 ± 0.07	0.02 ± 0.02	0.01 ± 0.01	98.50 ± 1.04
contain.	0.05 ± 0.03	0.02 ± 0.02	0.10 ± 0.04	0.01 ± 0.01	2.49 ± 0.08	0.13 ± 0.05	89.67 ± 2.42
building	0.28 ± 0.07	7.00 ± 0.39	0.27 ± 0.06	0.01 ± 0.01	0.86 ± 0.11	25.18 ± 0.42	74.97 ± 1.26
Predicted % correct	93.79 ± 0.99	72.64 ± 1.02	89.10 ± 1.44	99.17 ± 0.67	74.65 ± 2.70	81.03 ± 0.89	80.05 ± 0.48

Table 5: Confusion Matrix for XCS not using hedges (average for testing folds)

	Predicted						
Actual	bulb	car wind.	headlamp	optic	containers	building wind.	Actual Correct %
bulb	9.91 ± 0.11	0.05 ± 0.03	0.29 ± 0.08	0.00 ± 0.00	0.02 ± 0.02	0.13 ± 0.05	95.35 ± 0.99
car w.	0.08 ± 0.04	16.82 ± 0.45	0.37 ± 0.08	0.01 ± 0.01	0.07 ± 0.04	7.05 ± 0.47	68.97 ± 1.85
headlamp	0.44 ± 0.10	0.16 ± 0.06	5.56 ± 0.15	0.00 ± 0.00	0.20 ± 0.06	0.04 ± 0.03	86.81 ± 2.13
optic	0.01 ± 0.01	0.00 ± 0.00	0.01 ± 0.01	2.38 ± 0.07	0.00 ± 0.00	0.00 ± 0.00	99.17 ± 0.83
contain.	0.04 ± 0.03	0.05 ± 0.03	0.13 ± 0.05	0.00 ± 0.00	2.48 ± 0.08	0.10 ± 0.04	89.33 ± 2.69
building	0.24 ± 0.06	7.95 ± 0.46	0.63 ± 0.10	0.00 ± 0.00	1.09 ± 0.14	23.69 ± 0.48	70.52 ± 1.44
Predicted % correct	93.05 ± 1.04	68.41 ± 1.11	81.78 ± 1.65	99.67 ± 0.46	69.97 ± 2.91	77.56 ± 0.95	76.05 ± 0.46

VII. Conclusions

This research has shown that the inclusion of hedges to facilitate finer granularity of linguistic fuzzy modelling improves the accuracy of the resulting systems. LF-XCS allows for a complete map to be modelled for a given problem. This offers the user not only rules to decide on classes but also rules to decide which classes to discard. In addition, this work provides additional insight into the relationships amongst the input data and the classes, thereby enabling a better understanding of the underlying problem which is being modelled. LF-XCS also leads to full linguistic classifiers, which are easily interpretable, and employ words defined by the user. Additionally, this approach inherits the capability from XCS of dealing with the issue of class imbalance, a very challenging practical matter for applications where classes may be underrepresented in the training data.

Whilst the initial results as reported herein are very promising, a number of important issues regarding the LF-XCS approach remain to be further investigated. These include how sensitive the work may be to the prescribed fuzzification, both in terms of the definition of original fuzzy set membership functions and of the number of these fuzzy sets, and how well it may cope with datasets of high dimensionality. For the former issue, more experiments are needed to evaluate the robustness property of the approach (over different datasets as well as varying the linguistic term set). For the latter, which would be rather difficult to resolve completely, simply by modifying the approach itself, a possible solution would be to include a data preprocessing procedure prior to the actual modelling process. Work on fuzzy feature selection [14] seems to provide a helpful start point for this.

Acknowledgements

This work has been developed while the first author was visiting Aberystwyth University thanks to the sponsorship of the Fundación Séneca, Murcia, Spain. The authors are grateful to the Sistemas Inteligentes Research Group of the Universidad de Murcia and the Advanced Reasoning Group of Aberystwyth University, and especially to Gregorio Martínez Pérez and Neil Mac Parthalain, for their support in this work, while taking full responsibility for the views expressed in this paper. Thanks also go to Dr. Grzegorz Zadora, of the Institute of Forensic Research in Krakow, Poland, for providing the real glass identification dataset, and to the reviewers for their helpful comments on this paper.

References

[1] S. Abe and R. Thawonmas. A fuzzy classifier with ellipsoidal regions. *IEEE Transactions on Fuzzy Systems*, 5:358–368, August 1997.

[2] H. R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5):724–740, 1992.

[3] J. C. Bezdek and S. K. Pal. *Fuzzy Models for Pattern Recognition: Methods that Search for Structures in Data*. IEEE Press, 1992.

[4] Andrea Bonarini. An introduction to learning fuzzy classifier systems. In *IWLCS*, volume 1813 of *Lecture Notes in Computer Science*, pages 83–106. Springer, 1999.

[5] Martin Butz, Tim Kovacs, Pier Luca Lanzi, and Stewart W. Wilson. Toward a theory of generalization and

learning in XCS. *IEEE Trans. on Evolutionary Computation*, 8(1):28–46, February 2004.

[6] Martin V. Butz and Stewart W. Wilson. An algorithmic description of XCS. *Journal of Soft Computing*, 6(3–4):144–153, 2002.

[7] Jorge Casillas, Brian Carse, and Larry Bull. Fuzzy-xcs: A michigan genetic fuzzy system. *IEEE Trans. on Fuzzy Systems*, 15(4):536–550, August 2007.

[8] P. R. Cohen. *Empirical Methods for Artificial Intelligence*. MII Press, Cambridge, 1995.

[9] Oscar Cordón and Francisco Herrera. A two-stage evolutionary process for designing tsk rule-based systems. *IEEE Trans. on Systems, Man and Cybernetics-Part B*, 29(6):703–715, December 1999.

[10] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA., 1989.

[11] A. F. Gómez Skarmeta and F. Jimenez. Fuzzy modeling with hybrid systems. *Fuzzy Sets and Systems*, 104(2):199–208, 1999.

[12] John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. *Induction: processes of inference, learning, and discovery*. MIT Press, 1986.

[13] J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Matlab Curriculum. Prentice Hall, 1997.

[14] R. Jensen and Q. Shen. *Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches*. IEEE Press and Wiley & Sons, 2008.

[15] Tim Kovacs. *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Distinguished Dissertations. Springer, 2004.

[16] A. Lozowski, T. J. Cholewo, and J. M. Zurada. Crisp rule extraction from perceptron network classifiers. In *Proceedings of International Conference on Neural Networks*, volume Plenary, Panel and Special Sessions, pages 94–99, Washington, D.C., 1996.

[17] Javier G. Marín-Blázquez and Gregorio Martínez Pérez. Intrusion detection using a linguistic hedged fuzzy-XCS classifier system. *Soft computing*, 13(3):273–290, 2009.

[18] Javier G. Marín-Blázquez, Gregorio Martínez Pérez, and Manuel Gil Pérez. A linguistic fuzzy-XCS classifier system. In *FUZZIEEE 2007*, pages 1–6, London, July 2007.

[19] Javier G. Marín-Blázquez and Qiang Shen. Linguistic hedges on trapezoidal fuzzy sets: A revisit. In *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, volume 1, pages 412–415, December 2001.

[20] Javier G. Marín-Blázquez and Qiang Shen. From approximative to descriptive fuzzy classifiers. *IEEE Trans. on Fuzzy Systems*, 10:484–497, August 2002.

[21] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

[22] Albert Orriols-Puig and Ester Bernadó-Mansilla. Bounding XCS's parameters for unbalanced datasets. In *GECCO 2006:*, volume 2, pages 1561–1568, 8-12 July 2006.

[23] Witold Pedrycz and Fernando Gomide. *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press, 1998.

[24] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–220, July 1959.

[25] S. F. Smith. *A learning system based on genetic adaptive algorithms*. University of Pittsburgh, 1980.

[26] Christopher Stone and Larry Bull. For real! XCS with continuous-valued inputs. *Evolutionary Computation*, 11(3):298–336, 2003.

[27] M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on Fuzzy Systems*, 1:7–31, February 1993.

[28] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[29] J. Valente de Oliveira. Semantic constrains for membership function optimization. *IEEE Trans. on Systems, Man and Cybernetics - Part A*, 29(1):128–138, Jan. 1999.

[30] L. X. Wang and J. M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, November-December 1992.

[31] Stewart W. Wilson. Classifier Systems Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

[32] Stewart W. Wilson. Get real! XCS with continuous-valued inputs. In *IWLCS*, volume 1813 of *Lecture Notes in Computer Science*, pages 209–222. Springer, 1999.

[33] Lofti A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning I. *Information Sciences*, 8:199–249, 1975.

[34] Lofti A. Zadeh. Fuzzy logic = computing with words. *IEEE Trans. on Fuzzy Systems*, 4(2):103–111, May 1996.

[35] Grzegorz Zadora. Glass analysis for forensic purposes – a comparison of classification methods. *Journal of Chemometrics*, 32(5–6):174–186, March 2007.